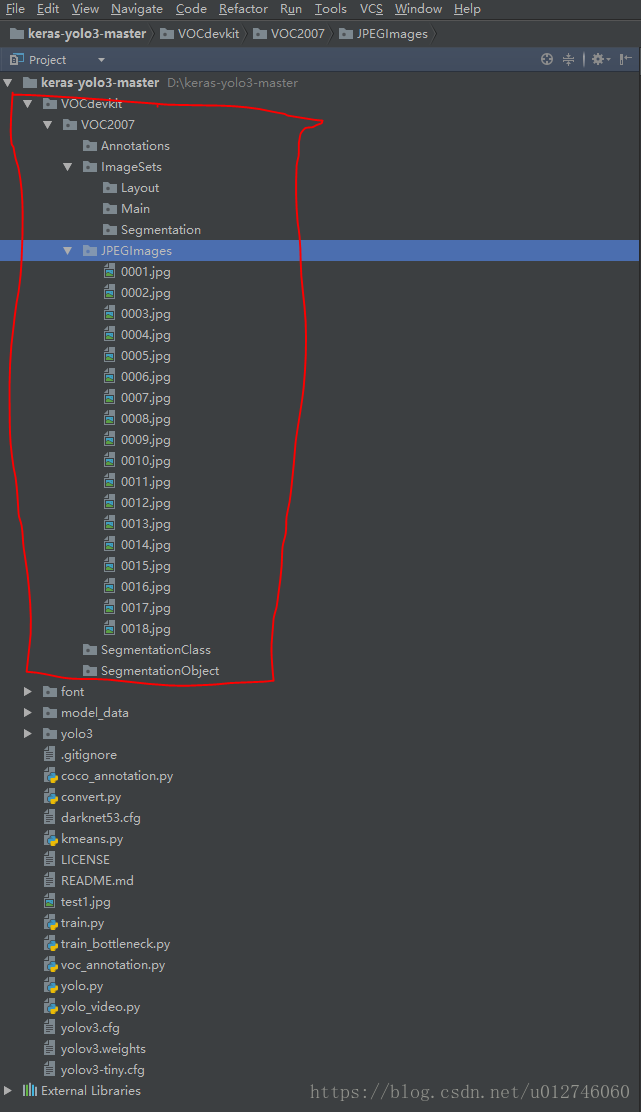
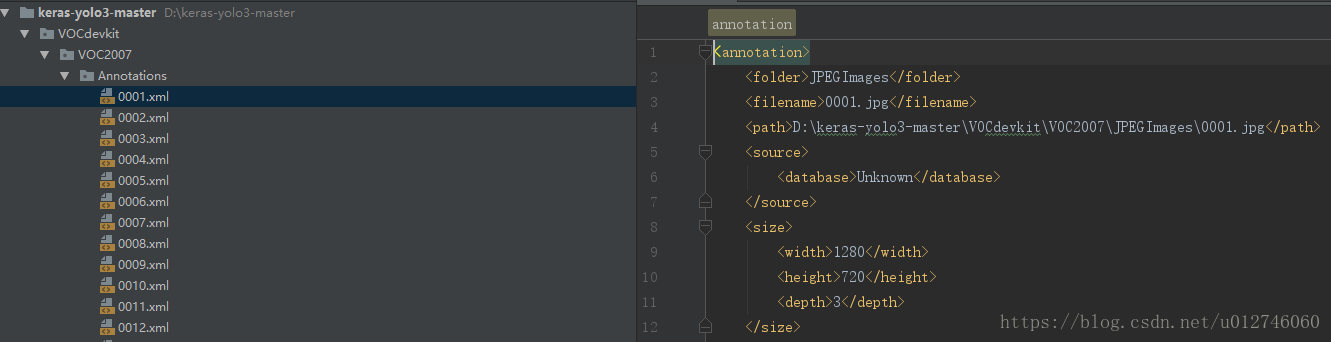
训练自己的数据集进行目标检测

          1、在工程下新建一个文件夹VOCdevkit，目录结构为VOCdevkit/VOC2007/,在下面就是新建几个默认名字的文件夹           Annotation，ImageSet（该目录还有三个文件需要建立），JPEGImages(把你所有的图片都复制到该目录里面，如下图），SegmentationClass，SegmentationObject。



           2、生成Annotation下的文件，安装工具labelImg。安装过程可参照：

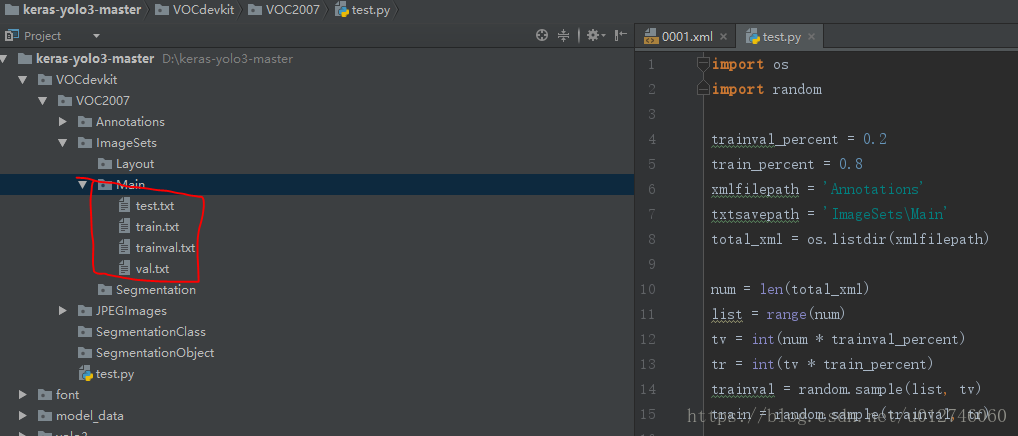
<https://blog.csdn.net/u012746060/article/details/81016993>，结果如下图：



            3、生成ImageSet/Main/4个文件。在VOC2007下新建一个python文件，复制如下代码

import os
import random
trainval\_percent = 0.2
train\_percent = 0.8
xmlfilepath = 'Annotations'
txtsavepath = 'ImageSets\Main'
total\_xml = os.listdir(xmlfilepath)
num = len(total\_xml)
list = range(num)
tv = int(num \* trainval\_percent)
tr = int(tv \* train\_percent)
trainval = random.sample(list, tv)
train = random.sample(trainval, tr)
ftrainval = open('ImageSets/Main/trainval.txt', 'w')
ftest = open('ImageSets/Main/test.txt', 'w')
ftrain = open('ImageSets/Main/train.txt', 'w')
fval = open('ImageSets/Main/val.txt', 'w')
for i in list:
name = total\_xml[i][:-4] + '\n'
if i in trainval:
ftrainval.write(name)
if i in train:
ftest.write(name)
else:
fval.write(name)
else:
ftrain.write(name)
ftrainval.close()
ftrain.close()
fval.close()
ftest.close()

                 运行代码之后，生成如下文件，VOC2007数据集制作完成。

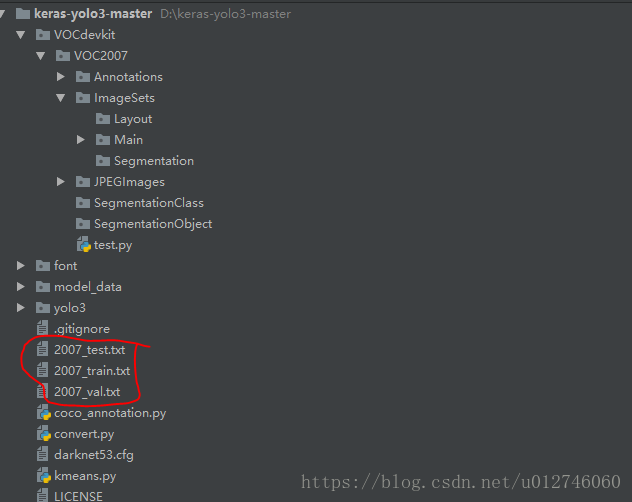


                4、生成yolo3所需的train.txt,val.txt,test.txt

                  生成的数据集不能供yolov3直接使用。需要运行voc\_annotation.py ，classes以检测两个类为例（车和人腿），在voc\_annotation.py需改你的数据集为：

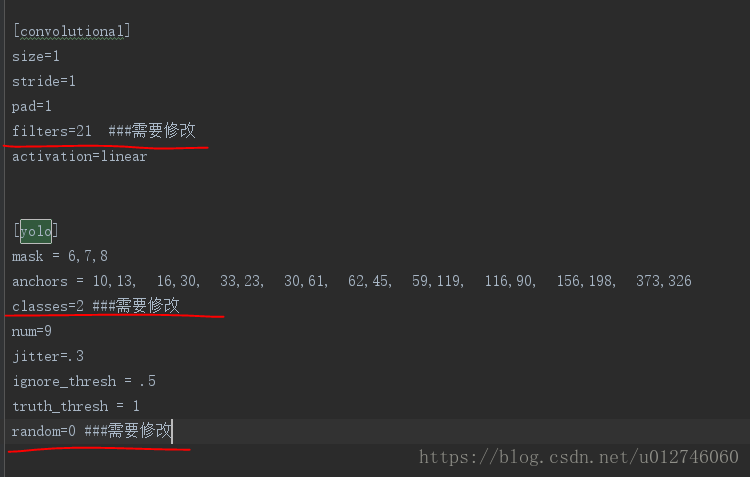
C6BBC14FC0AC4E5FAC280B6399814204.png

                运行之后，生成如下三个文件：



                5、修改参数文件yolo3.cfg

                 打开yolo3.cfg文件。搜索yolo(共出现三次)，每次按下图都要修改

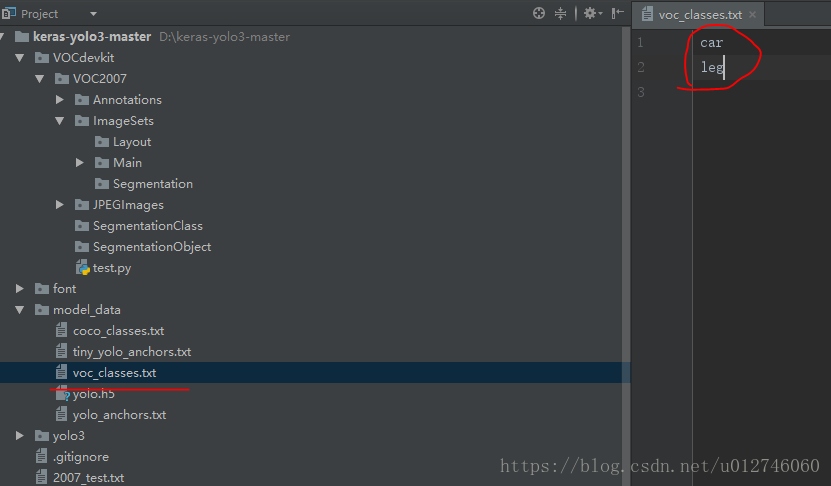


                  filter：3\*（5+len（classes））

                  classes:你要训练的类别数（我这里是训练两类）

                  random：原来是1，显存小改为0

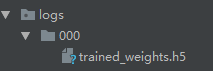
                 6、修改model\_data下的voc\_classes.txt为自己训练的类别



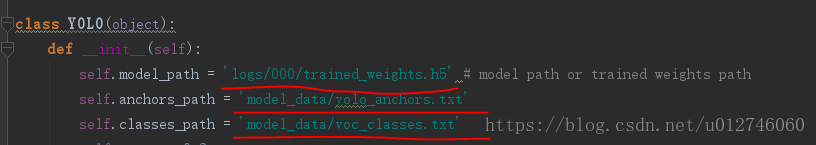
                7、 修改train.py代码（用下面代码直接替换原来的代码）

"""
Retrain the YOLO model for your own dataset.
"""
import numpy as np
import keras.backend as K
from keras.layers import Input, Lambda
from keras.models import Model
from keras.callbacks import TensorBoard, ModelCheckpoint, EarlyStopping
from yolo3.model import preprocess\_true\_boxes, yolo\_body, tiny\_yolo\_body, yolo\_loss
from yolo3.utils import get\_random\_data
def \_main():
annotation\_path = '2007\_train.txt'
log\_dir = 'logs/000/'
classes\_path = 'model\_data/voc\_classes.txt'
anchors\_path = 'model\_data/yolo\_anchors.txt'
class\_names = get\_classes(classes\_path)
anchors = get\_anchors(anchors\_path)
input\_shape = (416,416) # multiple of 32, hw
model = create\_model(input\_shape, anchors, len(class\_names) )
train(model, annotation\_path, input\_shape, anchors, len(class\_names), log\_dir=log\_dir)
def train(model, annotation\_path, input\_shape, anchors, num\_classes, log\_dir='logs/'):
model.compile(optimizer='adam', loss={
'yolo\_loss': lambda y\_true, y\_pred: y\_pred})
logging = TensorBoard(log\_dir=log\_dir)
checkpoint = ModelCheckpoint(log\_dir + "ep{epoch:03d}-loss{loss:.3f}-val\_loss{val\_loss:.3f}.h5",
monitor='val\_loss', save\_weights\_only=True, save\_best\_only=True, period=1)
batch\_size = 10
val\_split = 0.1
with open(annotation\_path) as f:
lines = f.readlines()
np.random.shuffle(lines)
num\_val = int(len(lines)\*val\_split)
num\_train = len(lines) - num\_val
print('Train on {} samples, val on {} samples, with batch size {}.'.format(num\_train, num\_val, batch\_size))
model.fit\_generator(data\_generator\_wrap(lines[:num\_train], batch\_size, input\_shape, anchors, num\_classes),
steps\_per\_epoch=max(1, num\_train//batch\_size),
validation\_data=data\_generator\_wrap(lines[num\_train:], batch\_size, input\_shape, anchors, num\_classes),
validation\_steps=max(1, num\_val//batch\_size),
epochs=500,
initial\_epoch=0)
model.save\_weights(log\_dir + 'trained\_weights.h5')
def get\_classes(classes\_path):
with open(classes\_path) as f:
class\_names = f.readlines()
class\_names = [c.strip() for c in class\_names]
return class\_names
def get\_anchors(anchors\_path):
with open(anchors\_path) as f:
anchors = f.readline()
anchors = [float(x) for x in anchors.split(',')]
return np.array(anchors).reshape(-1, 2)
def create\_model(input\_shape, anchors, num\_classes, load\_pretrained=False, freeze\_body=False,
weights\_path='model\_data/yolo\_weights.h5'):
K.clear\_session() # get a new session
image\_input = Input(shape=(None, None, 3))
h, w = input\_shape
num\_anchors = len(anchors)
y\_true = [Input(shape=(h//{0:32, 1:16, 2:8}[l], w//{0:32, 1:16, 2:8}[l], \
num\_anchors//3, num\_classes+5)) for l in range(3)]
model\_body = yolo\_body(image\_input, num\_anchors//3, num\_classes)
print('Create YOLOv3 model with {} anchors and {} classes.'.format(num\_anchors, num\_classes))
if load\_pretrained:
model\_body.load\_weights(weights\_path, by\_name=True, skip\_mismatch=True)
print('Load weights {}.'.format(weights\_path))
if freeze\_body:
# Do not freeze 3 output layers.
num = len(model\_body.layers)-7
for i in range(num): model\_body.layers[i].trainable = False
print('Freeze the first {} layers of total {} layers.'.format(num, len(model\_body.layers)))
model\_loss = Lambda(yolo\_loss, output\_shape=(1,), name='yolo\_loss',
arguments={'anchors': anchors, 'num\_classes': num\_classes, 'ignore\_thresh': 0.5})(
[\*model\_body.output, \*y\_true])
model = Model([model\_body.input, \*y\_true], model\_loss)
return model
def data\_generator(annotation\_lines, batch\_size, input\_shape, anchors, num\_classes):
n = len(annotation\_lines)
np.random.shuffle(annotation\_lines)
i = 0
while True:
image\_data = []
box\_data = []
for b in range(batch\_size):
i %= n
image, box = get\_random\_data(annotation\_lines[i], input\_shape, random=True)
image\_data.append(image)
box\_data.append(box)
i += 1
image\_data = np.array(image\_data)
box\_data = np.array(box\_data)
y\_true = preprocess\_true\_boxes(box\_data, input\_shape, anchors, num\_classes)
yield [image\_data, \*y\_true], np.zeros(batch\_size)
def data\_generator\_wrap(annotation\_lines, batch\_size, input\_shape, anchors, num\_classes):
n = len(annotation\_lines)
if n==0 or batch\_size<=0:returnNone
returndata\_generator(annotation\_lines, batch\_size, input\_shape, anchors, num\_classes)
if\_\_name\_\_ == '\_\_main\_\_':
\_main()

                   替换完成后，**千万千万值得注意的是**，因为程序中有logs/000/目录，你需要创建这样一个目录，这个目录的作用就是存放自己的数据集训练得到的模型。不然程序运行到最后会因为找不到该路径而发生错误。生成的模型trained\_weights.h5如下：



               8、修改yolo.py文件，如下将self这三行修改为各自对应的路径。



                   运行python yolo.py,输入自己要检测的类的图片即可查看训练效果了。

